# Reservoir Splatting for Temporal Path Resampling and Motion Blur Supplemental Document

JEFFREY LIU, University of Illinois Urbana-Champaign, USA
DAQI LIN, NVIDIA, USA
MARKUS KETTUNEN, NVIDIA, Finland
CHRIS WYMAN, NVIDIA, USA
RAVI RAMAMOORTHI, NVIDIA and UC San Diego, USA

CCS Concepts: • **Computing methodologies → Rendering**.

Additional Key Words and Phrases: real-time ray tracing, resampled importance sampling, ReSTIR, antialiasing, depth of field, motion blur

**ACM Reference Format:**
Jeffrey Liu, Daqi Lin, Markus Kettunen, Chris Wyman, and Ravi Ramamoorthi. 2025. Reservoir Splatting for Temporal Path Resampling and Motion Blur Supplemental Document. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3721238.3730646

This document explains derivations and details that did not fit in the main paper. In this document, we refer to prior-frame $\hat{p}$ as $\hat{p}^P$.

Section 1 derives MIS weights for reservoir splatting for general pixel filters, including the box filter as a special case. Section 2 then derives MIS weights for reservoir splatting with a backup sample, also including the box filter as a special case. Section 3 discusses additional implementation details for splatting and motion blur.

## 1 MIS WEIGHTS FOR RESERVOIR SPLATTING

In this section, we derive the MIS weights for scatter reuse *without* a backup sample.

Our inputs are $X_1, \ldots, X_N$ in the prior frame with the reprojection shift $T$ and the current pixel's canonical initial sample $Y^*$. Confidence weights are, in the same order, $c_1, \ldots, c_N$, and $c^*$. We will now derive formulas for $m_i(Y_i)$ and $m^*(Y^*)$ with the generalized balance heuristic.

The generalized balance heuristic [Lin et al. 2022] is

$$m_i(x) = \frac{c_i\ \hat{p}_{\leftarrow i}(x)}{\sum_{k=1}^{M} c_k\ \hat{p}_{\leftarrow k}(x)}, \tag{1}$$

where the "$\hat{p}$ from" function is

$$\hat{p}_{\leftarrow i}(y) = \hat{p}_i(T_i^{-1}(y)) \left|\partial T_i^{-1}/\partial y\right|, \tag{2}$$

and zero if $T_i^{-1}(y)$ is undefined or outside supp $X_i$.

We first derive $m_i(Y_i)$ by substitution into Equation 1. We get

$$m_i(Y_i) = \frac{c_i\ \hat{p}_{\leftarrow i}(Y_i)}{c^*\ \hat{p}_*(Y_i) + \sum_{k=1}^{N} c_k\ \hat{p}_{\leftarrow k}(Y_i)} \tag{3}$$

$$= \frac{c_i\ \hat{p}_i(T^{-1}(Y_i)) \left|\frac{\partial T^{-1}}{\partial Y_i}\right|}{c^*\ \hat{p}_*(Y_i) + \sum_{k=1}^{N} c_k\ \hat{p}_k(T^{-1}(Y_i)) \left|\frac{\partial T^{-1}}{\partial Y_i}\right|}. \tag{4}$$

Noting that $X_i = T^{-1}(Y_i)$ and $\hat{p}_i = h_i\ \hat{p}$,

$$m_i(Y_i) = \frac{c_i\ h_i(X_i)\hat{p}^P(X_i) \left|\frac{\partial Y_i}{\partial X_i}\right|^{-1}}{c^*\ h_*(Y_i)\hat{p}(Y_i) + \left(\sum_{k=1}^{N} c_k\ h_k(X_i)\right) \hat{p}^P(X_i) \left|\frac{\partial Y_i}{\partial X_i}\right|^{-1}}. \tag{5}$$

Next, we derive $m^*(Y^*)$ similarly. We get

$$m^*(Y^*) = \frac{c^*\ \hat{p}_*(Y^*)}{c^*\ \hat{p}_*(Y^*) + \sum_{k=1}^{N} c_k\ \hat{p}_{\leftarrow k}(Y^*)} \tag{6}$$

$$= \frac{c^*\ \hat{p}_*(Y^*)}{c^*\ \hat{p}_*(Y^*) + \sum_{k=1}^{N} c_k\ \hat{p}_k(T^{-1}(Y^*)) \left|\frac{\partial T^{-1}}{\partial Y^*}\right|}. \tag{7}$$

Denoting $X^* = T^{-1}(Y^*)$, we get

$$m^*(Y^*) = \frac{c^*\ h_*(Y^*)\hat{p}(Y^*)}{c^*\ h_*(Y^*)\hat{p}(Y^*) + \left(\sum_{k=1}^{N} c_k\ h_k(X^*)\right) \hat{p}^P(X^*) \left|\frac{\partial X^*}{\partial Y^*}\right|}. \tag{8}$$

In all of the above, the denominator contains a confidence-weighted sum of pixel filter values of the reverse-splatted sample.

Next, we derive simplified formulas for the box filter. A sample's resampling weight $w$ can be positive only if the sample is in the current pixel. This is always true for $Y^*$, and for $Y_i$ we find the contributing indices by scattering.

First, we assume $Y_i$ is defined and in the current pixel. We start from Equation 5 and set $h_i(X_i) = 1$ since $X_i$ comes from pixel $i$ and $h_*(Y_i) = 1$ by assumption that $Y_i$ is in the current pixel. The only contributing index in the sum is $i$, since $X_i$ is in pixel $i$, and $h_i(X_i) = 1$. We get

$$m_i(Y_i) = \frac{c_i\ \hat{p}^P(X_i) \left|\frac{\partial Y_i}{\partial X_i}\right|^{-1}}{c^*\ \hat{p}(Y_i) + c_i\ \hat{p}^P(X_i) \left|\frac{\partial Y_i}{\partial X_i}\right|^{-1}}. \tag{9}$$

Next, we simplify $m^*(Y^*)$ from Equation 8. The initial sample is always in the current pixel, so we set $h_*(Y^*) = 1$. Let $r$ be the prior-frame pixel $Y^*$ reverse-splats into. This is the only pixel contributing to the sum in the denominator, and $h_{r^*}(X^*) = 1$. We get

$$m^*(Y^*) = \frac{c^*\ \hat{p}(Y^*)}{c^*\ \hat{p}(Y^*) + c_r\ \hat{p}^P(X^*)\left|\frac{\partial X^*}{\partial Y^*}\right|}. \tag{10}$$

If the reverse splat $X^* = T^{-1}(Y^*)$ is undefined or does not land on any pixel, the term containing $X^*$ is removed.

## 2 MIS WEIGHTS WITH A BACKUP SAMPLE

In this section, we derive the MIS weights for scatter reuse *with a backup sample*. The derivation is similar to Section 1.

Our inputs are $X_1, \ldots, X_N$ in the prior frame with the reprojection shift $T$, a backup sample $X^b$ in the prior frame with shift mapping $T_b$, and the current pixel's canonical initial sample $Y^*$. Their confidence weights are, in the same order, $c_1, \ldots, c_N$, $c^b$, and $c^*$. We will now derive formulas for $m_i(Y_i)$, $m^b(Y^b)$, and $m^*(Y^*)$ with the generalized balance heuristic.

We first derive $m_i(Y_i)$ by substitution into Equation 1. We get

$$m_i(Y_i) = \frac{c_i\ \hat{p}_{\leftarrow i}(Y_i)}{c^*\ \hat{p}_*(Y_i) + c^b\ \hat{p}_{\leftarrow b}(Y_i) + \sum_{k=1}^N c_k\ \hat{p}_{\leftarrow k}(Y_i)} \tag{11}$$

$$= \frac{c_i\ \hat{p}_i(T^{-1}(Y_i))\left|\frac{\partial T^{-1}}{\partial Y_i}\right|}{c^*\ \hat{p}_*(Y_i) + c^b\ \hat{p}_b(T_b^{-1}(Y_i))\left|\frac{\partial T_b^{-1}}{\partial Y_i}\right| + \sum_{k=1}^N c_k\ \hat{p}_k(T^{-1}(Y_i))\left|\frac{\partial T^{-1}}{\partial Y_i}\right|}. \tag{12}$$

Denoting $Z_i = T_b^{-1}(Y_i)$, noting $X_i = T^{-1}(Y_i)$, and $\hat{p}_i = h_i\,\hat{p}$,

$$m_i(Y_i) = \frac{c_i\ h_i(X_i)\hat{p}^P(X_i)\left|\frac{\partial Y_i}{\partial X_i}\right|^{-1}}{c^*\ h_*(Y_i)\hat{p}(Y_i) + c^b\ h_b(Z_i)\hat{p}^P(Z_i)\left|\frac{\partial Z_i}{\partial Y_i}\right| + \left(\sum_{k=1}^N c_k\ h_k(X_i)\right)\hat{p}^P(X_i)\left|\frac{\partial Y_i}{\partial X_i}\right|^{-1}}. \tag{13}$$

Next, we derive $m^*(Y^*)$ similarly. We get

$$m^*(Y^*) = \frac{c^*\ \hat{p}_*(Y^*)}{c^*\ \hat{p}_*(Y^*) + c^b\ \hat{p}_{\leftarrow b}(Y^*) + \sum_{k=1}^N c_k\ \hat{p}_{\leftarrow k}(Y^*)} \tag{14}$$

$$= \frac{c^*\ \hat{p}_*(Y^*)}{c^*\ \hat{p}_*(Y^*) + c^b\ \hat{p}_b(T_b^{-1}(Y^*))\left|\frac{\partial T_b^{-1}}{\partial Y^*}\right| + \sum_{k=1}^N c_k\ \hat{p}_k(T^{-1}(Y^*))\left|\frac{\partial T^{-1}}{\partial Y^*}\right|}. \tag{15}$$

Denoting $X^* = T^{-1}(Y^*)$ and $Z^* = T_b^{-1}(Y^*)$, we get

$$m^*(Y^*) = \frac{c^*\ h_*(Y^*)\hat{p}(Y^*)}{c^*\ h_*(Y^*)\hat{p}(Y^*) + c^b\ h_b(Z^*)\hat{p}^P(Z^*)\left|\frac{\partial Z^*}{\partial Y^*}\right| + \left(\sum_{k=1}^N c_k\ h_k(X^*)\right)\hat{p}^P(X^*)\left|\frac{\partial X^*}{\partial Y^*}\right|}. \tag{16}$$

Finally, we derive $m^b(Y^b)$, with $Y^b = T_b(X^b)$:

$$m^b(Y^b) = \frac{c^b\ \hat{p}_{\leftarrow b}(Y^b)}{c^*\ \hat{p}_*(Y^b) + c^b\ \hat{p}_{\leftarrow b}(Y^b) + \sum_{k=1}^N c_k\ \hat{p}_{\leftarrow k}(Y^b)} \tag{17}$$

$$= \frac{c^b\ \hat{p}_b(T_b^{-1}(Y^b))\left|\frac{\partial T_b^{-1}}{\partial Y^b}\right|}{c^*\ \hat{p}_*(Y^b) + c^b\ \hat{p}_b(T_b^{-1}(Y^b))\left|\frac{\partial T_b^{-1}}{\partial Y^b}\right| + \sum_{k=1}^N c_k\ \hat{p}_k(T^{-1}(Y^b))\left|\frac{\partial T^{-1}}{\partial Y^b}\right|}. \tag{18}$$

With $Y^b = T_b(X^b)$, denoting $Z^b = T^{-1}(X^b)$, we have

$$m^b(Y^b) = \frac{c^b\ h_b(X^b)\hat{p}^P(X^b)\left|\frac{\partial Y^b}{\partial X^b}\right|^{-1}}{c^*\ h_*(Y^b)\hat{p}(Y^b) + c^b\ h_b(X^b)\hat{p}^P(X^b)\left|\frac{\partial Y^b}{\partial X^b}\right|^{-1} + \left(\sum_{k=1}^N c_k\ h_k(Z^b)\right)\hat{p}^P(Z^b)\left|\frac{\partial Z^b}{\partial X^b}\right|}. \tag{19}$$

In all of the above, the denominator contains a confidence-weighted sum of pixel filter values at the reverse-splatted sample.

Next, we derive simplified formulas for the box filter. A sample's resampling weight $w$ can be positive only if the sample is in the current pixel. This is always true for $Y^*$, for $Y^b$ by construction when the backup pixel $b$ exists, and for $Y_i$ we find the contributing indices by scattering.

First, we assume $Y_i$ is defined and in the current pixel. We start from Equation 13 and set $h_i(X_i) = 1$ since $X_i$ comes from pixel $i$, $h_*(Y_i) = 1$ by assumption that $Y_i$ is in the current pixel, and $h_b(Z_i) = 1$ since $T_b^{-1}$ shifts from the current pixel to the backup pixel. The only contributing index in the sum is $i$, since $X_i$ is in pixel $i$, and $h_i(X_i) = 1$. We get

$$m_i(Y_i) = \frac{c_i\ \hat{p}^P(X_i)\left|\frac{\partial Y_i}{\partial X_i}\right|^{-1}}{c^*\ \hat{p}(Y_i) + c^b\ \hat{p}^P(Z_i)\left|\frac{\partial Z_i}{\partial Y_i}\right| + c_i\ \hat{p}^P(X_i)\left|\frac{\partial Y_i}{\partial X_i}\right|^{-1}}. \tag{20}$$

If the shift $Z_i = T_b^{-1}(Y_i)$ fails, the term involving $Z_i$ is removed.

Next, we simplify $m^*(Y^*)$ from Equation 16. The initial sample is always in the current pixel, so we set $h_*(Y^*) = 1$. Since $Z^* = T_b^{-1}(Y^*)$ is in the backup pixel, we have $h_b(Z^*) = 1$. Let $r^*$ be the prior-frame pixel $Y^*$ reverse-splats into. This is the only pixel contributing to the sum in the denominator, and $h_{r^*}(X^*) = 1$. We get

$$m^*(Y^*) = \frac{c^*\ \hat{p}(Y^*)}{c^*\ \hat{p}(Y^*) + c^b\ \hat{p}^P(Z^*)\left|\frac{\partial Z^*}{\partial Y^*}\right| + c_{r^*}\ \hat{p}^P(X^*)\left|\frac{\partial X^*}{\partial Y^*}\right|}. \tag{21}$$

If the reverse splat $X^* = T^{-1}(Y^*)$ is undefined or does not land on any pixel, the term containing $X^*$ is removed. If the shift $Z^* = T_b^{-1}(Y^*)$ into the backup pixel fails, or the backup pixel does not exist, the term containing $Z^*$ is removed.

Finally, we simplify $m^b(Y^b)$ from Equation 19. Assuming we get to evaluate $m^b(Y^b)$, the backup pixel must exist, and so $h_b(X^b) = 1$. The shift $Y^b = T_b(X^b)$ must have been defined, so $h_*(Y^b) = 1$. Finally, let $r^b$ be the pixel of the reverse-splat $Z^b = T^{-1}(Y^b)$. This is the only contributing pixel in the sum in the denominator. We get

$$m^b(Y^b) = \frac{c^b\, \hat{p}^P(X^b) \left|\frac{\partial Y^b}{\partial X^b}\right|^{-1}}{c^*\, \hat{p}(Y^b) + c^b\, \hat{p}^P(X^b) \left|\frac{\partial Y^b}{\partial X^b}\right|^{-1} + c_{r^b}\, \hat{p}^P(Z^b) \left|\frac{\partial Z^b}{\partial X^b}\right|}. \quad (22)$$

If $Z^b$ is undefined or lands outside the image (i.e., $r^b$ does not exist), the term containing $Z^b$ is removed.

## 3 IMPLEMENTATION

This section discusses certain implementation details of our algorithm that did not fit into the main paper.

### 3.1 Splatting Operation

In our implementation, we use the box filter. Conceptually, all $N = H \times W$ prior-frame reservoirs may splat into a single pixel, but it is impossible to allocate a buffer storing a local array of $N$ entries per current-frame pixel. In practice, each frame will splat at most $N$ reservoirs, so we instead allocate a *global* array of $N$ entries, mapping prior-frame reservoirs to its current-frame pixel.

The splatting operation is then implemented using atomic counters: one counter per pixel and one global counter. We run a set of shaders in the following sequence:

(1) *Forward-Projection:* a thread per previous-frame pixel to forward-project its reservoir to the current frame and increment the corresponding pixel $j$'s atomic counter, which returns an offset in the "local" array of splatted samples corresponding to pixel $j$. Once finished, the per-pixel atomic counters store the length of each current-frame pixel's array.
(2) *Sorting:* a thread per current-frame pixel to increment the global atomic counter by its array length, which returns an offset in the global buffer for the "local" array to be stored.
(3) *Global Array Insertion:* a thread per previous-frame pixel to write its pixel index to the global buffer based on the global offset and the local array offset.
(4) *Resampling:* a thread per current-frame pixel to loop through all associated previous-frame samples, to shift and resample.

Note that the actual shift mapping on previous-frame pixels can be computed in step (1) to minimize thread divergence. The for-loop in step (4) can then simply execute the weighted reservoir resampling.

### 3.2 Confidence Weights in Temporal Resampling

In practice, we consider both the splatted reservoirs *and* the backup reservoir as temporal samples. To avoid over-downweighting the canonical sample, we replace $c^b, c_{r^b}, c_i, c_{r^*}$ from Section 2 with $\alpha c^b, \alpha c_{r^b}, (1-\alpha)c_i, (1-\alpha)c_{r^*}$ respectively, using $\alpha = 0.5$ for simplicity. We leave the exploration of $\alpha$ to future work.

### 3.3 Motion Blur

Our implementation completely decouples shutter time from frame time, i.e., we can simulate any shutter time $\tau$ along with any simulated frame time $\Delta t$.

*3.3.1 Camera Interpolation.* We need to shoot rays with arbitrary time samples $t \in [t_0^P, t_1^P)$ or $t \in [t_0, t_1)$ to account for time-involved shifts. To do so, we allocate a static array to serve as a circular buffer, storing camera orientations. At the beginning of each frame, the oldest entry is replaced by the new camera orientation. By tracking the head of the circular buffer, shaders can orient the camera at arbitrary $t$ by computing the two orientations that $t$ lies between. Using the relative offset between the two timestamps, the orientation at $t$ is computed by linearly interpolating the positions and the angle between the two forward vectors.

*3.3.2 Naïve Area ReSTIR Extension.* In our naïve Area ReSTIR extension, we use the motion vectors computed based on $t_1$ and $t_1^P$, i.e., the motion vectors provided by Falcor's V-buffer render pass [Kallweit et al. 2022]. Any other $t$ is still just an arbitrary choice, and should not be expected to perform any better. The inverse view-projection matrices for the camera orientations at $t_1^P, t_1$ are already provided, so it would require extra computation to compute the motion vectors at other timestamps.

## REFERENCES

Simon Kallweit, Petrik Clarberg, Craig Kolb, Tomáš Davidovič, Kai-Hwa Yao, Theresa Foley, Yong He, Lifan Wu, Lucy Chen, Tomas Akenine-Möller, Chris Wyman, Cyril Crassin, and Nir Benty. 2022. The Falcor Rendering Framework. https://github.com/NVIDIAGameWorks/Falcor [Online; accessed 22-August-2023].

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized Resampled Importance Sampling: Foundations of ReSTIR. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 75:1–75:23. https://doi.org/10.1145/3528223.3530158